CoMMEDIA: Separating Scaramouche from Harlequin to Accurately Estimate Items Frequency in Distributed Data Streams

Emmanuelle Anceaume, Yann Busnel

CoMMEDIA: Separating Scaramouche from Harlequin to Accurately Estimate Items Frequency in Distributed Data Streams

Emmanuelle Anceaume[1] and Yann Busnel[2⋆]

[1] IRISA / CNRS Rennes (France), emmanuelle.anceaume@irisa.fr
[2] LINA / Université de Nantes (France), Yann.Busnel@univ-nantes.fr

Abstract. In this paper, we investigate the problem of estimating the number of times data items that recur in very large distributed data streams. We present an alternative approach to the well-known Count-Min Sketch in order to reduce the impact of collisions on the accuracy of the estimation. We propose to decrease, for each concerned item, the over-estimation that results from these collisions. Our sketch, called CoMMEDIETTA, keeps track of the most frequent items of the stream, and removes their weight from the one of the items with which these frequent items collide. By doing so, we significantly improve upon the Count-Min Sketch by achieving a randomized (ε, δ)-approximation algorithm. We then propose to judiciously distribute this local sketch to estimate the global frequency of any item that may recur in multiple streams. This distributed sketch, called CoMMEDIA (for Count-Min Sketch-based Estimation of Data Items Arrival frequency), organizes nodes of the system in a distributed hash table (DHT) such that each node implements a tiny local sketch on a reduced number of items. By doing so we guarantee a significantly more accurate estimation of item frequencies. Simulations both on synthetic and real traces confirm the accuracy of CoMMEDIA.

Keywords: Distributed data streams, approximation algorithm, data items frequency distribution, decentralized data structures.

1 Introduction

Large-scale distributed networks and huge data flows have become the norm in the last decade. Handling large-scale networks has led to the construction of various kinds of efficient overlay networks. These overlays essentially allow nodes to reach any destination in the network using a sub linear number of hops in the size of the network. Data flows bring a huge amount of information that need the design of online algorithms to accurately estimate statistics.

Two main approaches exist to monitor in real time massive data streams. The first one consists in regularly sampling the input streams so that only a

⋆ **Contact author:** LINA / Université de Nantes – 2, rue de la Houssinière – BP 92208 – 44322 Nantes Cedex 03 – France – Fax: +33 (0)2 51 12 58 12

limited amount of data items is locally kept [1, 2]. This allows to exactly compute functions on these samples. However, accuracy of this computation fully depends on the volume of data that has been sampled and their locations in the stream. Worse, an adversary can easily take advantage of the sampling policy to hide its packets among the other packets of the stream. The second approach consists in scanning each piece of data of the input stream on the fly, and in locally keeping compact synopses or sketches that contain the most important information about data items. Actually, a rich body of algorithms and techniques have been proposed for the past several years to efficiently compute statistics on massive data streams. These include the computation of the number of different data items in a given stream [3–5], the frequency moments [6], the most frequent data items [6, 7], or the entropy of the stream [8, 9]. In particular, estimating the number of times items recur in data streams is a very often sought statistics as it allows, for example, to efficiently detect worms and deny of service attacks in intrusion detection services, or to perform in real time traffic engineering in network applications. The *Count-Min Sketch* [10] is the synopsis which so far has been proven to be the best one in terms of space and time performance to estimate data item frequency. Briefly, the Count-Min Sketch (abbreviated as CM sketch in the following) is a fixed size array of $t \times k$ counters associated with t pairwise independent hash functions, one per line of the array. Each hash function h_i maps items of the stream uniformly onto $1, \ldots, k$. Each time an item v is read from the input stream, the corresponding location under h_i, for $i = 1, \ldots, t$, is incremented by some constant value. Upon receipt of a request to know the current frequency of a given item v in the stream, CM returns the minimum value among the t concerned entries of the array.

However, because k is typically much smaller than items identifier space, hash collisions do occur. This affects the accuracy of the estimation when items frequently recur in the input stream, that is when the size m of the input stream becomes large. To circumvent this saturation issue Dimitropoulos et al [11] keep only fresh items of the input stream so that the number of distinct items that appear in the sketch does not increase with the size of the stream. While effective to decrease the number of collusions, such an approach opens the door to adversarial behaviors, and in particular dormant attacks, where the adversary from time to time floods the stream with its own items, eclipsing accordingly most of the other items in the sketch.

In this paper, we present an alternative approach to reduce the impact of collisions on the accuracy of the estimation. We propose to decrease, for each concerned item, the over-estimation that results from these collisions. Our sketch, called in the following CoMMEDIETTA, keeps track of the most frequent items of the stream, and removes their weight from the one of the items with which these frequent items collide. By doing so, we significantly improve upon the Count-Min Sketch by achieving a randomized (ε, δ)-approximation algorithm (see Section 2.3). We then propose to judiciously distribute this local sketch to estimate the global frequency of any item that may recur in multiple streams. This distributed algorithm, called CoMMEDIA (for Count-Min Sketch-based Es-

timation of Data Items Arrival frequency), organizes nodes of the system in a distributed hash table (DHT) such that each node implements a tiny local sketch on a reduced number of items. By doing so we guarantee a significantly more accurate estimation of item frequencies. Finally, simulations both on synthetic and real traces that exhibit skews in various ways confirm the accuracy of CoMMEDIA.

The remaining of the paper is organized as follows. Section 2 presents the model this work relies on. Section 3 describes and analyzes CoMMEDIETTA, the local sketch that accurately estimates item frequencies. Section 4 presents a simple way to distribute sketches among all the nodes to so that distributed streams can be efficiently monitored both in time and space. Section 5 evaluates the minimum effort the adversary needs to exert to bias items frequency estimation. Finally, extended simulations have been conducted in different adversarial contexts and the main lessons drawn from these simulations are presented in Section 6, while Section 7 concludes.

2 Model and Background

2.1 Model

We present the computation model under which we analyze our algorithms. We consider a large scale system $\mathcal{S}$, such that each node $i \in \mathcal{S}$ receives a large sequence $\sigma^{(i)}$ of data items or symbols $< u, v, w, \ldots >$ drawn from a large universe N (these data items can for example model TCP/IP packets, HTTP requests [12]). For a given stream $\sigma^{(i)}, i \in \mathcal{S}$, we denote by $m^{(i)}$ the size of $\sigma^{(i)}$ that is the number of items in it and $n^{(i)}$ the number of distinct items. The number of times item u appears in a stream is called the frequency of item u, and is denoted by f_u. Items arrive regularly and quickly, and due to memory constraints, items must be processed sequentially and in an online manner. In the following we only focus on the frequency estimations of the items that effectively appear in the stream [10]. We refer the reader to [13] for a detailed description of data streaming models and algorithms.

2.2 Adversary

We assume the presence of malicious (*i.e.*, Byzantine) nodes that collectively try to subvert the system by manipulating the prescribed protocol. We model these adversarial behaviors through an adversary that fully controls and manipulates these malicious nodes. We suppose that the adversary is strong in the sense that it may actively tamper with the data stream of any node i by observing, and injecting a potentially large number ℓ of items. Indeed, the goal of the adversary is to judiciously increase the frequency of its ℓ items to bias the frequency estimation of the items generated by correct nodes. The number ℓ is chosen by the adversary according to the parameters of the sketches at correct nodes. By *correct*, we mean a node present in the system which is not malicious. Note that

correct nodes cannot *a priori* distinguish items sent by correct nodes from the ones sent by malicious ones. Classically, we assume that the adversary can neither drop a message exchanged between two correct nodes nor tamper with its content without being detected. This is achieved by assuming the existence of a signature scheme (and the corresponding public-key infrastructure) ensuring the authenticity and integrity of messages. This refers to the authenticated Byzantine failure model [14]. We finally suppose that any algorithm run by any correct node is public knowledge to avoid some kind of security by obscurity. However the adversary has not access to the local random coins used in the algorithms.

2.3 Preliminaries

We first present notations and background on data streams analysis that make this paper self-contained.

2-universal Hash Functions In the following, we use hash functions randomly picked from a 2-universal hash functions family. A collection $\mathcal{H}$ of hash functions $h : \{1, \ldots, M\} \to \{0, \ldots, M'\}$ is said to be *2-universal* if for every two different items $x, y \in [M]$, $\mathbb{P}_{h \in \mathcal{H}}\{h(x) = h(y)\} \leq \frac{1}{M'}$, which is the probability of collision obtained if the hash function assigned truly random values to any $x \in [M]$.

Randomized (ε, δ)-approximation Algorithm A randomized algorithm $\mathcal{A}$ is said to be an (ε, δ)-approximation of a function ϕ on σ if for any sequence of items in the input stream σ, $\mathcal{A}$ outputs $\hat{\phi}$ such that $\mathbb{P}\{\mid \hat{\phi} - \phi \mid > \varepsilon\phi\} < \delta$, where $\varepsilon, \delta > 0$ are given as parameters of the algorithm.

3 The CoMMEDIETTA sketch

Prior to describing our distributed sketch algorithm (*cf.* Section 4), we describe and analyze the accuracy of the local sketch CoMMEDIETTA. We suppose that CoMMEDIETTA is fed with the input stream denoted by σ.

3.1 Building blocks

We first describe two algorithms that form the building blocks of CoMMEDIETTA. The first one, proposed by Cormode and Muthukrishnan [10], estimates items frequency of all the nodes of a stream. The second one is due to Misra and Gries [15] and allows to deterministically output most frequent items. Both algorithms have been designed in the stream data model (*cf.* Section 2.1).

Estimating the frequency of each data item For any item v in the input stream σ, the algorithm proposed by Cormode and Muthukrishnan [10] outputs an estimation $\hat{f}_v^{(\mathtt{CM})}$ of the number of times v has occurred in the stream so far, *i.e.*, the frequency of v. The error of the estimator in answering a query for $\hat{f}_v^{(\mathtt{CM})}$ is within an error $\varepsilon(m - f_v)$ with probability $1 - \delta$. The estimation is

Algorithm 1: Count-Min Sketch [10]

Input: An input stream σ; precision parameters k and t;
Output: The estimate $\hat{f}_v$ for the frequency of any item v read from the input stream

$C[1..t][1..k] \leftarrow 0$;
Choose t 2-universal hash functions $h_1, \ldots, h_t : \{1, \ldots, N\} \rightarrow \{1, \ldots, k\}$;
for $v \in \sigma$ **do**
  for $i = 1$ **to** t **do**
    $C[i][h_i(v)] \leftarrow C[i][h_i(v)] + 1$;
Upon query of $\hat{f}_v^{(\mathtt{CM})}$: **return** $\min_{1 \leq i \leq t} C[i][h_i(v)]$;

computed by maintaining a two-dimensional array C of $k \times t$ counters with $k = \lceil e/\varepsilon \rceil$ and $t = \lceil \log_2(1/\delta) \rceil$, and by using a collection of 2-universal hash functions $\{h_1, \ldots, h_t\}$. Each time an item v is read from the input stream, this causes one counter per line to be incremented. When a query is issued to get the estimate $\hat{f}_v^{(\mathtt{CM})}$, the returned value is equal to the minimum among the t values of $C[i][h_i(v)]$ $(1 \leq i \leq t)$. Algorithm 1 presents the pseudo-code of the Count-Min Sketch algorithm. The update time per element is significantly sub-linear in the size of the sketch [10].

Theorem 1 ([10]). *The Count-Min Sketch algorithm with parameters* $k = \lceil e/\varepsilon \rceil$ *and* $t = \lceil \log_2(1/\delta) \rceil$ *returns for any item* v *an estimate* $\hat{f}_v^{(\mathtt{CM})}$ *such that* $\mathbb{P}\{\mid \hat{f}_v^{(\mathtt{CM})} - f_v \mid > \varepsilon(m - f_v)\} < \delta$ *with* $\mathcal{O}(1/\varepsilon \log(1/\delta) \log m)$ *bits of space.*

Determining the most frequent data items The problem of determining the most frequent items in a stream has also been studied extensively in the data stream literature. Thereafter, we propose an enhanced version of the Misra and Gries [15] algorithm. The basic Misra and Gries algorithm is deterministic, and outputs the items that occur more than m/a times in a stream for a given a. It maintains a counters such that for each counter, its key is the item read from the stream and its value is related to the frequency of items. Initially, all the counters are set to $(\perp, 0)$. Afterwards, when an item is read from the stream, if that item has already a counter associated to it, then this counter is incremented. If this is not the case and if there are still free counters available, then one of these free counters is allocated to this new item and its value is set to 1. Otherwise, all the allocated counters are decremented by one, and if after this operation some of them are equal to 0 then their keys are erased and the counters are released. We improve upon this algorithm by augmenting each counter with a second value that allows to wait as most as possible before removing an item from A. In some sense, this second value keeps track of the randomness with which items are received in the stream. The pseudo-code of this enhanced Misra Gries algorithm is presented in Algorithm 2.

Algorithm 2: Enhanced Misra-Gries

Input: An input stream σ; a precision parameter a;
Output: a frequent items in σ and an estimate of their frequency

```
for j ∈ [0..a] do A[j] ← (⊥, ⊥, ⊥);
for v ∈ σ do
    if ∃j such that A[j] = (v, −, −) then Increment the first value of A[j];
    else
        if ∃j such that A[j] = (⊥, ⊥, ⊥) then A[j] = (v, 1, 0);
        else
            if ∃j such that A[j] = (−, κ, κ) with κ ∈ ℕ then
                Pick j′ at random among {j | A[j] = (−, κ, κ), κ ∈ ℕ};
                A[j′] = (v, 0, 1);
            else
                for j = 1 to a do Increment the second value of A[j];
Upon query of f̂_v^(MG): if ∃j such that A[j] = (v, −, −) then return first value of
A[j] else return 0;
```

Theorem 2. *The enhanced Misra and Gries algorithm with parameter a returns for any item v an estimate $\hat{f}_v^{(\mathtt{MG})}$ such that $f_v - \frac{m(a-H)}{a(a+2-H)} \leq \hat{f}_v^{(\mathtt{MG})} \leq f_v$ with $\mathcal{O}(a(\log m + \log n))$ bits of space, where H corresponds to the number of heavy hitters (*i.e.*, items with frequency greater than m/a).*

Proof. Due to space constraints, the proof of this theorem, largely inspired from [15], is provided in Appendix A.

Note that the lower bound of $\hat{f}_v^{(\mathtt{MG})}$ is reached with a very low probability as it requires that item v be replaced by a new arriving one in A each time it is possible. This happens with a probability in the order of magnitude of $a^{-m/a}$. The interested reader is invited to consult the proof in Appendix A.

3.2 The CoMMEDIETTA sketch: splitting Scaramouche from Harlequin

As said in the Introduction, in CM Sketch, heavy hitters significantly bias the estimation of the frequency of other items with which heavy hitters collide. This mays lead to a noticeable over-estimation of their frequency. As will be shown in Section 5, this feature can be fully exploited by the adversary to bias items frequency estimation. By simply removing the weight of heavy hitters from the one of the items with which these frequent items collide, CoMMEDIETTA improves upon the CM Sketch by being a (ε, δ)-approximation algorithm.

Pseudo-code of CoMMEDIETTA is presented in Algorithm 3. It consists of two phases. The first one is executed upon receipt of item $u, v, \ldots$ of the stream, while the second one returns the frequency estimation of any item u. Instead of

Algorithm 3: CoMMEDIETTA

Input: An input stream σ, precision parameters k, t and a;
Output: Estimation of the frequency of any item v in σ

Initialize data structures A and C as respectively presented in Algorithms 1 and 2;
for $v \in \sigma$ **do**
  Task T_1 **:**
    Update sketch C with v as described in Algorihm 1;
  Task T_2 **:**
    Update sketch A with v as described in Algorithm 2;
Upon query of $\hat{f}_u$: **return** $\min_{1 \leq i \leq t} \left(C[i][h_i(u)] - \sum_{v \in A \setminus \{u\},\ h_i(v)=h_i(u)} \hat{f}_v^{(\texttt{MG})} \right)$;

directly returning the minimum among the t values of $C[i][h_i(u)]$ $(1 \leq i \leq t)$ as done in the Count-Min Sketch, the weight imposed by highly frequent items that collide with u are removed from the t involved counters in C. It is important to remark that both CoMMEDIETTA CM Sketch cannot prevent queries for items that have not appeared in the stream so far to be issued. Of course, the returned values do not make any sense. However this does not jeopardize correctness of both sketches (see Theorem 3) as frequency estimations are guaranteed solely for all the items that have appeared in the stream.

3.3 Analysis of CoMMEDIETTA

In this section, we prove that for any ε and δ, and for any data item u read from the input stream, CoMMEDIETTA gives an (ε, δ)-approximation of f_u.

Theorem 3. *CoMMEDIETTA, run with parameters $k = \lceil e/\varepsilon \rceil$, $t = \lceil \log_2(1/\delta) \rceil$ and $a = \lceil \varepsilon n(1 - 1/m) \rceil$, returns for any item u an estimate $\hat{f}_u$ such that $\mathbb{P}\left\{ \hat{f}_u - f_u \geq \varepsilon f_u \right\} \leq \delta$ using $O\left(\frac{1}{\varepsilon} \log \frac{1}{\delta} \log m + \varepsilon n \left(1 - \frac{1}{m}\right) (\log m + \log n) \right)$ bits of space.*

Proof. From algorithm 3, the estimation $\hat{f}_u$ item u overestimates its real frequency f_u. Indeed, every counter $C[i][h_i(u)]$ is equal to the sum of the exact frequencies of all data item that collide with u, that is that share the same hashed value as u (*i.e.*, all $v \in \sigma$ such that $h_i(v) = h_i(u)$). Moreover, the enhanced Misra-Gries algorithm underestimates the frequency of heavy hitters, *i.e.*, $\hat{f}_v^{(\texttt{MG})} \leq f_v$ (see Theorem 2). Thus, as $\hat{f}_v^{(\texttt{MG})} = 0$ for any $v \notin A$, we have

$$\begin{aligned} \hat{f}_u &= \min_{i \in \{1,\ldots,t\}} \left(C[i][h_i(u)] - \sum_{v \in A \setminus \{u\}} \hat{f}_v^{(\texttt{MG})} \mathbf{1}_{h_i(v)=h_i(u)} \right) \\ &= \min_{i \in \{1,\ldots,t\}} \left(f_u + \sum_{v \in \mathcal{N} \setminus \{u\}} \left(f_v - \hat{f}_v^{(\texttt{MG})} \right) \mathbf{1}_{h_i(v)=h_i(u)} \right) \geq f_u. \end{aligned} \tag{1}$$

Let X_u be the random variable that represents the excess of $\hat{f}_u$ with respect to f_u. From Relation (1), let us first analyze the excess of a given $i \in \{1, \ldots, t\}$. We denote by $X_{u,i}$ the random variable that measures this specific excess. We have $X_{u,i} = \sum_{v \in N \setminus \{u\}} \left(f_v - \hat{f}_v^{(\texttt{MG})} \right) \mathbf{1}_{h_i(v)=h_i(u)}$ and $X_u = \min_{i \in \{1,\ldots,t\}} X_{u,i}$.

By the 2-universality property of the family from which hash function h_i is drawn, we have $\mathbb{E}[h_i(v) = h_i(u)] \leq 1/k$. Thus, by linearity of the expectation,we get that

$$\mathbb{E}[X_{u,i}] = \sum_{v \in N \setminus \{u\}} \mathbb{E}\left[\left(f_v - \hat{f}_v^{(\texttt{MG})} \right) \mathbf{1}_{h_i(v)=h_i(u)} \right] \leq \frac{m - f_u}{k} - \frac{\sum_{v \in N \setminus \{u\}} \mathbb{E}\left[\hat{f}_v^{(\texttt{MG})} \right]}{k}. \tag{2}$$

As k increases, the error made by the Count-Min Sketch diminishes, which imposes less constraints on the accuracy that must be guaranteed by the enhanced Misra-Gries algorithm. As $\hat{f}_v^{(\texttt{MG})} = 0$ for any $v \notin A$ and $\hat{f}_v^{(\texttt{MG})} \geq f_v - m(a - H)/(a^2 + 2a - aH)$ (see Theorem 2), we have

$$\sum_{v \in N \setminus \{u\}} \hat{f}_v^{(\texttt{MG})} = \sum_{v \in A \setminus \{u\}} \hat{f}_v^{(\texttt{MG})} \geq \sum_{v \in A \setminus \{u\}} f_v - m \frac{aH - H^2}{a^2 + 2a - aH}.$$

By Algorithm 2, data structure A contains heavy hitters. Thus the lower bound for $\sum_{v \in A \setminus \{u\}} f_v$ is reached in presence of a uniform stream. We then have $\min \sum_{v \in A \setminus \{u\}} f_v = (a-1)m/n$. By choosing $a = \lceil \varepsilon n (1 - 1/m) \rceil$ and $k = \lceil e/\varepsilon \rceil$, and from $f_u \geq 1$ for every u, it is easily checked that $\frac{\sum_{v \in N \setminus \{u\}} \hat{f}_v^{(\texttt{MG})}}{k} \geq \frac{m - 2f_u}{k}$.

From Relation (2), we get that

$$\mathbb{E}[X_{u,i}] \leq \frac{m - f_u}{k} - \frac{m - 2f_u}{k} = \frac{f_u}{k}.$$

Since for every u in the stream, $f_u \geq 1$ and $X_{u,i} \geq 0$ for any $i \in \{1, \ldots, t\}$, we can apply the Markov's inequality. Moreover, as $k = \lceil e/\varepsilon \rceil$, we get

$$\mathbb{P}\left\{ X_{u,i} \geq \varepsilon f_u \right\} \leq \frac{\mathbb{E}[X_{u,i}]}{\varepsilon f_u} \leq \frac{f_u}{k \varepsilon f_u} \leq \frac{1}{2}. \tag{3}$$

Relation (3) holds for any $i \in \{1, \ldots, t\}$. The CoMMEDIETTA sketch owns t such estimators, mutually independent. By Relation 1, we are able to estimate the excess of CoMMEDIETTA as the minimum of $X_{u,i}$ over all $i \in \{1, \ldots, t\}$. Then, we obtain

$$\begin{aligned} \mathbb{P}\left\{ \hat{f}_u - f_u \geq \varepsilon f_u \right\} &= \mathbb{P}\left\{ \min_{i \in \{1,\ldots,t\}} X_{u,i} \geq \varepsilon f_u \right\} \\ &= \mathbb{P}\left\{ X_{u,1} \geq \varepsilon f_u, \ldots, X_{u,t} \geq \varepsilon f_u \right\} \\ &= \prod_{i \in \{1,\ldots,t\}} \mathbb{P}\left\{ X_{u,i} \geq \varepsilon f_u \right\} \leq \frac{1}{2^t} \leq \delta \end{aligned}$$

by definition of t, which concludes the proof. □

4 CoMMEDIA: a distributed sketch for frequency estimation

In the previous section, we have presented CoMMEDIETTA, a sketch that gives for any ε and any δ an (ε, δ)-approximation of items frequency, for any item received in an input stream. We are now ready to present CoMMEDIA, a distributed sketch that allows to estimate the global frequency of items that may recur in distributed streams. CoMMEDIETTA combines the features of both the streaming model and large scale distributed overlays. As in the streaming model, the input is read on the fly and processed with a minimum workspace and time. As in large scale overlays, the load of the system is partitioned among all the nodes of the system. Specifically, all the nodes of the system self-organize in a structured overlay. Structured overlays, also called Distributed Hash Tables (DHTs), build their topology according to structured graphs. For most of them, the following principles hold: The identifier space is partitioned among all the nodes of the overlay. Nodes self-organize within the graph according to a distance function D based on nodes identifiers (*e.g.*, two nodes are neighbors if their identifiers share some common prefix), plus possibly other criteria such as geographical distance. Each application-specific object, or data-item, is assigned a unique identifier, called *key*, which is the hashed value of the object. In our context, we consider that this hash function is picked among the 2-universal family. Both keys and node identifiers belong to the same identifier space. Each node owns a fraction of all the data items of the system. The mapping derives from the distance function D. Following the seminal work of Plaxton et al [16], diverse DHTs have been proposed (such as CAN [17], Chord [18], D2B [19]). All these DHTs have been proven to be highly satisfactory in terms of both efficiency and scalability (their key-based routing mechanism guarantees operations whose complexity in messages and latency scale logarithmically with system size).

These principles directly apply to our context. All the nodes of $\mathcal{S}$ self-organize in a DHT, and are responsible for the frequency estimation of all the items whose key are closer to them according to the distance D implemented in the DHT. Let $\mathcal{I}_i$ be the set of items whose keys are close to node i than to any other nodes, and $\sigma^{(i)}$ denote the input stream of node i. Then, each node $i \in \mathcal{S}$ locally maintains a CoMMEDIETTA sketch that solely estimates the frequency of all the items that belong to $\mathcal{I}_i$. For all the other items v that recur in $\sigma^{(i)}$ but that do not belong to $\mathcal{I}_i$, node i routes them to node j such that $v \in \mathcal{I}_j$. Finally, a query for getting any item v frequency estimation simply consists in sending this query to the (single) node in charge of v. As will be shown in Theorem 4, this construction first guarantees the (ε, δ)-approximation of items frequency, and second is space-efficient. Indeed, the total workspace needed by CoMMEDIA (that is the sum of each CoMMEDIETTA space implemented at each node $i \in \mathcal{S}$) is strictly equal to the workspace that would be needed by a single CoMMEDIETTA to estimate the frequency of all the items recurring in the union of all the distributed streams. These results are very interesting. First they show that by judiciously splitting the estimation of item frequency over the nodes of the system, one keeps the good performance of a (ε, δ)-approximation algorithm.

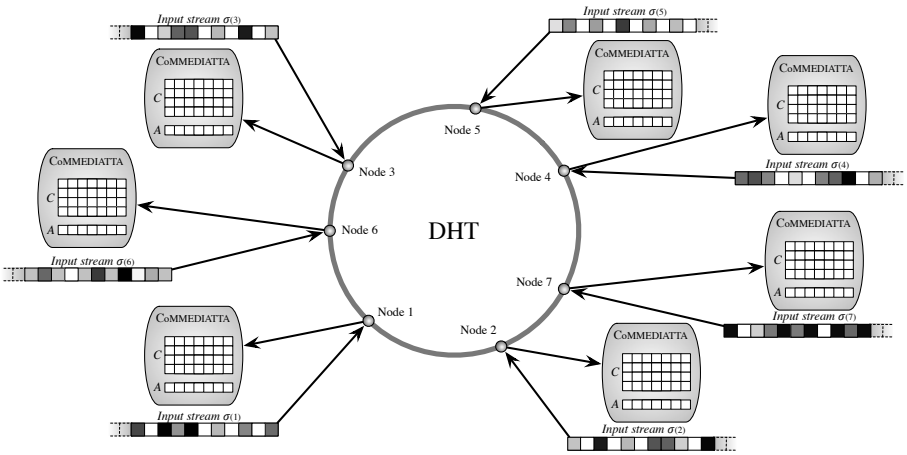

Fig. 1: CoMMEDIA structure, where each node runs a CoMMEDIETTA instance over a DHT.

Second it shows that doing statistics over distributed streams does not bring any additional cost with respect to a centralized approach in terms of space at the expense of communication costs. Indeed, each time a node i receives in its input stream an item v that does not belong to $\mathcal{I}_i$, item v must be forwarded to the node j such that j is closer to v key. Note that in expectation this is achieved in a logarithmic number of hops in the size of $\mathcal{S}$.

Theorem 4. *Executing CoMMEDIA on s nodes, with parameters $k = \lceil e/(s\varepsilon) \rceil$, $t = \lceil \log_2(1/\delta) \rceil$ and $a = \lceil \varepsilon n(1 - 1/m)/s \rceil$ for each CoMMEDIETTA instance, returns for any item u an estimate $\hat{f}_u$ such that $\mathbb{P}\left\{ \hat{f}_u - f_u \geq \varepsilon f_u \right\} \leq \delta$ using $O\left(\frac{1}{s\varepsilon} \log \frac{1}{\delta} \log m + \frac{\varepsilon n}{s}\left(1 - \frac{1}{m}\right)(\log m + \log n)\right)$ bits of space per node (where $m = \sum_{i \in \mathcal{S}} m^{(i)}$ and n represents the number of distinct data items that occur on all streams $\sigma^{(i)}$).*

Proof. Let h_{DHT} be the the hash function used for the key assignment in the DHT. By the properties of hash functions, the key space can be split into s mutually exclusive subsets forming a partition of $\mathcal{N}$. Suppose now that all the CoMMEDIETTA Count-Min arrays, spread over the s nodes, are concatenated into a huge unique array whose size is equal to $\frac{1}{\varepsilon} \times \log \frac{1}{\delta}$. The probability that any two items u and v share the same cell on each line is equal to the probability that both u and v are assigned to the same node (which is lower than $1/s$ by the 2-Universality properties of h_{DHT}) and that they share the same cell on this node (which is lower than $1/k$). By assumption, the t hash functions are pairwise independent. By assumption of the theorem, $k = \lceil e/(s\varepsilon) \rceil$. Thus this probability is lower than or equal to ε/e, which corresponds to the one of a unique CoMMEDIETTA instance with parameter $k = \lceil e/\varepsilon \rceil$ and $t = \lceil \log_2(1/\delta) \rceil$.

Let us focus on the enhanced Misra-Gries algorithm ran on each node in $\mathcal{S}$ with parameter $a = \lceil (1-1/m)\varepsilon n/s \rceil$. Each node in charge of n/s items receives a stream of size m/s in average. Thus according to Theorem 2, for every u in the any stream, we have

$$f_u \geq \hat{f}_u^{(\text{MG})} \geq f_u - \frac{\frac{m}{s}\left(\frac{a}{s} - \frac{H}{s}\right)}{\frac{a}{s}\left(\frac{a}{s} + 2 - \frac{H}{s}\right)} = f_u - \frac{m(a-H)}{a(a+2s-H)} \geq f_u - \frac{m(a-H)}{a(a+2-H)}.$$

Thus the guarantee brought by the union of all instances of the enhanced Misra-Gries on each node in $\mathcal{S}$ is strictly equal to the one guaranteed by a single CoMMEDIETTA instance with parameter $a = \lceil \varepsilon n(1-1/m) \rceil$.

Direct application of Theorem 3 concludes the proof. □

5 Effort needed by the adversary to subvert CoMMEDIA

As previously said, we suppose that the adversary has enough resources to generate a large number of items, and to judiciously inject them in the input stream $\sigma^{(i)}$ of any correct node $i \in \mathcal{S}$, so that items frequency are over-estimated.

From Algorithm 3, this can be only achieved by increasing the error made on the estimations $\hat{f}_v^{(\text{CM})}$ of item v as by Theorem 2 we have that $f_u - \frac{m(a-H)}{a(a+1)} \leq \hat{f}_u^{(\text{MG})}$ holds with probability 1. Thus to disrupt the estimation $\hat{f}_u^{(\text{CM})}$ of any item u, the adversary has to generate sufficiently many items $v_1, \ldots, v_\ell$ such that for all the lines s, $s = 1, \ldots, t$ of C array, there exists an item v_j such that $h_s(v_j) = h_s(u)$. Recall that the t hash functions are locally chosen, thus the adversary cannot know which identifiers map to $h_1(u), \ldots, h_t(u)$. By injecting numerous times these items $v_1, \ldots, v_\ell$, the estimation $\hat{f}_u^{(\text{CM})}$ will be arbitrarily overestimated. Note that the adversary will blindly bias the frequency estimation of many items, including its owns.

By conducting an analysis similar to the one achieved in [20], we can derive the minimum effort that needs to be exerted by the adversary to make its attack successful with probability $1 - \eta$, where $\eta < 1$. This is achieved by modeling an attack as a urn problem, where each entry of C is modeled as an urn and each received distinct item as a ball. N_ℓ is the random variable that counts the number of non empty urns among any set of k urns at time ℓ. Let U_k be the number of balls needed in order to obtain all the k urns occupied, *i.e.*, with at least one ball. It is easily checked that $\mathbb{P}\{U_1 = 1\} = 1$ and that, for $\ell \geq k \geq 2$, we have

$$U_k = \ell \Longrightarrow N_{\ell-1} = k - 1.$$

From [20], we get, for $k \geq 2$ and $\ell \geq k$,

$$\mathbb{P}\{U_k = \ell\} = \frac{1}{k^{\ell-1}} \sum_{r=0}^{k-1} (-1)^r \binom{k-1}{r} (k-1-r)^{\ell-1}.$$

Finally, we consider the integer E_k which counts the number of balls needed to get a collision in all the $k \times t$ urns. Note that this number is independent of t as

Table 1: Key values of E_k

k	10 ($\varepsilon \sim 0.3$)		50 ($\varepsilon \sim 0.05$)		250 ($\varepsilon \sim 0.01$)	
η	10^{-1}	10^{-4}	10^{-1}	10^{-4}	10^{-1}	10^{-4}
E_k	44	110	306	651	$1,617$	$3,363$

Table 2: Statistics of real data traces.

Data trace	items	distinct	max. freq.
NASA (Jul.)	1,891,715	81,983	17,572
NASA (Aug.)	1,569,898	75,058	6,530
ClarkNet (Aug.)	1,654,929	90,516	6,075
ClarkNet (Sep.)	1,673,794	94,787	7,239
Saskatchewan	2,408,625	162,523	52,695

by definition, the t experiments in parallel are identical and independent. Thus, filling entirely a set of k urns leads to obtain all the t sets of k urns occupied. For given value of k and $\eta \in (0,1)$, integer E_k is defined by

$$E_k = \inf \left\{ \ell \geq k \left| \sum_{i=k}^{\ell} \mathbb{P}\{U_k = i\} > 1 - \eta \right. \right\}. \tag{4}$$

Recall that parameters k of Algorithm 3 are common knowledge (except the random local coins) and thus the adversary is capable of deriving E_k according to the desired probability η. Finally, the adversary needs to inject in the input stream $E_k + a$ items to bias the frequency estimation of all the items that have been received in the input streams.

The main results of this analysis are summarized in Table 1. The most important one is that the effort that needs to be exerted by the adversary to subvert the sampling service can be made arbitrarily large by any correct node by just increasing the memory space of the sampler. The second one, which derives from the first one, is the absence of relationship between the effort of the adversary and the size of the population size.

6 Performance Evaluation of CoMMEDIA

In this section, we have conducted experiments to evaluate the CoMMEDIA quality to estimate the frequency of items that recur in distributed streams.

Settings of the Experiments. We have implemented CoMMEDIA and run a series of experiments on different types of streams and for different parameters settings. We have fed our algorithm with both real-world data sets and synthetic traces that have been distributed over the nodes of the system. Real data give a realistic representation of some existing systems, while the latter ones allow to capture phenomenon which may be difficult to obtain from real-world traces, and thus allow to check the robustness of our strategies. We have varied all the significant parameters of our algorithm, that is the number of distinct data items n in each stream, the size of the enhanced Misra-Gries memory a, the number k of entries in each line of the Count-Min matrix, and the number t of lines of this matrix. For each parameters setting, we have conducted and averaged 100 trials of the same experiment, leading to a total of more than $10,000,000$ experiments for the evaluation of our algorithm. Real data have been downloaded

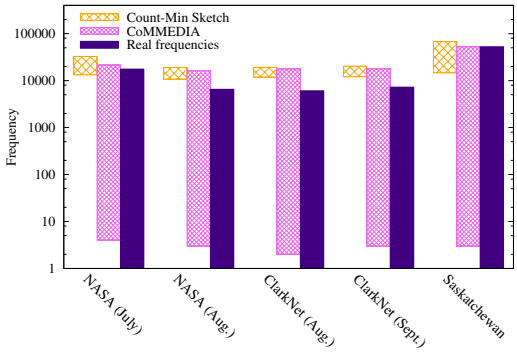

Fig. 2: Estimation of the range of frequencies of items that recur in real data traces as computed by the Count-Min sketch and the CoMMEDIA sketch compared to their real frequency. We have $\varepsilon = 10^{-2}$ and $\delta = 10^{-9}$.

from the repository of Internet network traffic [21]. We have used three large traces among the available ones, representing different periods of HTTP requests to WWW server of respectively NASA Kennedy Space Center, ClarkNet provider and University of Saskatchewan. Table 2 presents some statistics of these data traces, in term of stream size, population size in each stream and the number of occurrences of the most frequent item. Note that all these benchmarks share a Zipfian behavior, with a lower α parameter for the University of Saskatchwan.

Main Lessons drawn from the Experiments. We now present the main lessons drawn from these experiments. Due to space constrain, only a subset of all the conducted experiments are included in this paper.

Figure 2 shows the quality of both the Count-Min sketch and the CoMMEDIA one to estimate item frequency when fed with real traces. This figure confirms first that estimations always over-estimate the real frequency of items. Second, it shows that CoMMEDIA clearly improves upon the quality of Count-Min by closely estimating the full range of frequencies of items, even for very low frequencies.

Figure 3 represents a kind of isopleths in which the horizontal axis shows the size a of the enhanced Misra-Gries array, the vertical axis represents the data items, and the body of the graph depicts the frequency estimation of each item while using CoMMEDIA. A lighter color is representative of a highly frequent item. The figure on the left represents an attack, in which the adversary injects massively more than 100 fake data items while all the other items occur rarely and uniformly in the whole stream. The one on the right represents a classical stream distribution following a Zipf law (*i.e.*, no more than 20 items occur very frequently). Results got with Count-Min correspond to the extreme left of each isopleth (*i.e.*, when $a = 0$), while the real item frequencies are shown at their extreme right (*i.e.*, when $a = n = 1000$). Both isopleth at the top of Figure 3 have been obtained by using constant space with respect to the one used by Count-Min. That is, for each size $a = 0, \ldots, 1000$ of the enhanced Misra-Gries array,

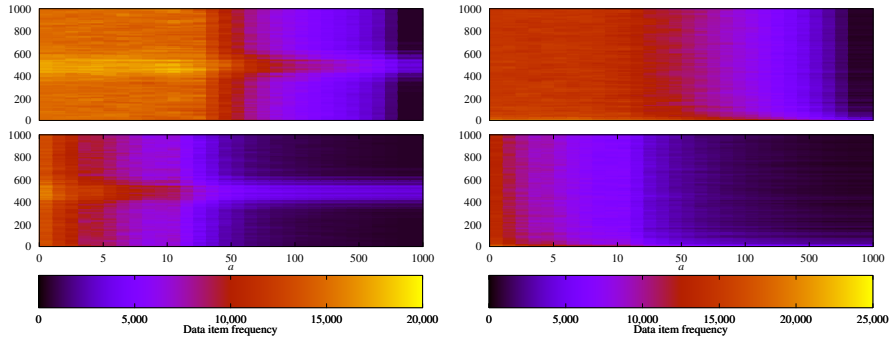

Fig. 3: Frequency distribution as a function of a. Settings: $m = 100,000$; $n = 1,000$; $\varepsilon = 0.1$; $\delta = 0.1$.

the size of the data structure C is diminished accordingly. On the other hand, for both isopleths at the bottom of the figure, the size of the Count-Min sketch is kept constant, and the size a of A data structure is incremented from $a = 0$ to $a = 1000$. The main result drawn from this figure is that both Count-Min and thus CoMMEDIA initially over-estimate all the frequencies which explains why the sum of all the estimates frequency greatly exceeds the size of the stream ($1000 \times 10,000 > m$). However, by increasing a little bit the used space for CoMMEDIA with respect to the one used for Count-Min, we obtain a significant improvement of the frequency estimation. Indeed with $a = 50$ ($k = 28$, and $t = 8$), we obtain a frequency estimation which is very close to the exact one. On the other hand, the above isopleths are not so impressive, nevertheless from $a = 50$, the estimation converges relatively fast to the exact ones.

7 Conclusion

In this paper we have investigating the problem of estimating on the fly the frequency at which items recur in distributed streams. Our approach combines tools and probabilistic algorithms from both the distributed community and the data streaming one. By doing so, we have first improved upon the Count-Min sketch which was, so far, the best sketch in terms of space and time performance to estimate data item frequency. We have then presented a solution to monitor distributed streams by distributing local sketches over the nodes of the system. We have proven that this distributed algorithm guarantees a (ε, δ)-approximation of item frequency estimation in a space-efficient way. This has been validated with experiments on both synthetic and real traces.

References

1. Subhabrata, B.K., Krishnamurthy, E., Sen, S., Zhang, Y., Chen, Y.: Sketch-based change detection: Methods, evaluation, and applications. In: Internet Measurement Conference. (2003) 234–247

2. Karamcheti, V., Geiger, D., Kedem, Z., Muthuskrishnan, S.: Detecting malicious network traffic using inverse distribution of packet contents. In: Proc. of the workshop on Mining Network Data (MineNet) co-located with ACM SICOMM. (2005)
3. Bar-Yossef, Z., Jayram, T.S., Kumar, R., Sivakumar, D., Trevisan, L.: Counting distinct elements in a data stream. In: Proc. of the 6th International Workshop on Randomization and Approximation Techniques (RANDOM), Springer-Verlag (2002) 1–10
4. Flajolet, P., Martin, G.N.: Probabilistic counting algorithms for data base applications. Journal of Computer and System Sciences **31**(2) (1985) 182–209
5. Kane, D.M., Nelson, J., Woodruff, D.P.: An optimal algorithm for the distinct element problem. In: Proc. of the Symposium on Principles of Databases (PODS). (2010)
6. Alon, N., Matias, Y., Szegedy, M.: The space complexity of approximating the frequency moments. In: Proc. of the 28th annual ACM symposium on Theory of computing (STOC). (1996) 20–29
7. Charikar, M., Chen, K., Farach-Colton, M.: Finding frequent items in data streams. Theoretical Computer Science **312**(1) (2004) 3–15
8. Chakrabarti, A., Cormode, G., McGregor, A.: A near-optimal algorithm for computing the entropy of a stream. In: ACM-SIAM Symposium on Discrete Algorithms. (2007) 328–335
9. Lall, A., Sekar, V., Ogihara, M., Xu, J., Zhang, H.: Data streaming algorithms for estimating entropy of network traffic. In: Proc. of the joint international conference on Measurement and modeling of computer systems (SIGMETRICS), ACM (2006)
10. Cormode, G., Muthukrishnan, S.: An improved data stream summary: The count-min sketch and its applications. Journal of Algorithms **55**(1) (2005) 58–75
11. Dimitropoulos, X., Stoecklin, M., Hurley, P., Kind, A.: The eternal sunshine of the sketch data structure. Computer Networks **52**(17) (2008)
12. Demaine, E.D., López-Ortiz, R., Munro, J.I.: Frequency estimation of internet packet streams with limited space. In: In Proceedings of the 10th Annual European Symposium on Algorithms, Springer-Verlag (2002) 348–360
13. Muthukrishnan: Data Streams: Algorithms and Applications. Now Publishers Inc. (2005)
14. Lynch, N.: Distributed Algorithms. Morgan Kaufmann Publishers (1996)
15. Misra, J., Gries, D.: Finding repeated elements. Science of Computer Programming **2**(2) (1982) 143–152
16. Plaxton, C.G., Rajaraman, R., Richa, A.W.: Accessing nearby copies of replicated objects in a distributed environment. In: Proceedings of the 9th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA). (1997)
17. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. In: Proceedings of the ACM SIGCOMM. (2001)
18. Stoica, I., Liben-Nowell, D., Morris, R., Karger, D., Dabek, F., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of the ACM SIGCOMM. (2001)
19. Fraigniaud, P., Gauron, P.: D2B: a de Bruijn based content-addressable network. Theoretical Computer Science **355**(1) (2006) 65–79
20. Anceaume, E., Busnel, Y., Sericola, B.: Uniform node sampling service robust against collusions of malicious nodes. In: Proc. of the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Budapest, Hungary (June 2013)
21. the Internet Traffic Archive: `http://ita.ee.lbl.gov/html/traces.html`. Lawrence Berkeley National Laboratory (April 2008)

A Appendix

Theorem 2. *The enhanced Misra and Gries algorithm with parameter a returns for any item u an estimate $\hat{f}_u^{(\mathtt{MG})}$ such that $f_u - \frac{m(a-H)}{a(a+2-H)} \leq \hat{f}_u^{(\mathtt{MG})} \leq f_u$ with $\mathcal{O}(a(\log m + \log n))$ bits of space, where H corresponds to the number of heavy hitters (*i.e.*, items with frequency greater than m/a).*

Proof. Consider the tuple array A used in Algorithm 2. Each entry of A stores a key (*i.e.*, an item) requiring $\lceil \log n \rceil$ bits, and two values that call for at most $\lceil \log m \rceil$ bits. Since there are at most a key/values tuple in A at any time, the total space required is $O(a(\log m + \log n))$ bits.

Consider now the quality of the Misra-Gries sketch. Let v be a given heavy hitter v in the stream (that is, $f_v \geq m/a$).

First of all, by the last line of Algorithm 2, each item that does not belong to A is associated with a 0 output. Moreover, the first value of v tuple is incremented only when an occurrence of v effectively occurs in the stream. Thus, we have $\hat{f}_u^{(\mathtt{MG})} \leq f_u$.

On the other hand, whenever the second value of v tuple is incremented, $a-1$ other second values are also incremented, corresponding to distinct items in the stream. Since the stream consists of m data items, there can be at most m/a such decrements. A grouping argument is used to argue that any item which occurs more than m/a times must be stored by the algorithm when it terminates [15]. Thus, in the best case scenario, the v tuple is never overwritten during the execution, and the first value of this tuple is exactly f_v.

Let us consider now the worst case scenario in which v tuple is overwritten regularly from A. In such a scenario, each time v is written in A, it will be replaced by another item as soon as possible. In order to overwrite v tuple from A, then $a-1$ other distinct values have to occur right after the insertion of v into A. Indeed, they need to fill all the possible a slots of A, plus another distinct item from the current keys of A to increment the second value of v tuple and finally, a last distinct one to enter in A at the exact place of v. Then, in order to "forget" about a unique occurrence of v , it requires a sequence of $a+2$ distinct items (including v) in the stream.

Let H denote the number of heavy hitters in the stream. More formally, we have $H = \mid \{u \mid f_u \geq m/a\} \mid$. Thus, as the stream consists of m items, we have at most $m - Hm/a$ slots for the above scenario to hold. The $c+2$ distinct items required for this scenario can obviously be picked among the heavy hitters. Then, the maximum occurrence of v that can be hiden using Algorithm 2 is given by

$$\frac{1}{a+2-H}\left(m - H\frac{m}{a}\right) = \frac{m}{a+2-H}\left(\frac{a-H}{a}\right) = \frac{m(a-H)}{a(a+2-H)}$$

that concludes the proof. □